Architecture for access to a compute-intensive image mosaic service in the NVO

G. Bruce Berriman, David Curkendall, John C. Good, Joseph C. Jacob, Daniel S. Katz, et al.

G. Bruce Berriman, David Curkendall, John C. Good, Joseph C. Jacob, Daniel S. Katz, Mihseh Kong, Serge Monkewitz, Reagan Moore, Thomas A. Prince, Roy E. Williams, "Architecture for access to a compute-intensive image mosaic service in the NVO," Proc. SPIE 4846, Virtual Observatories, (16 December 2002); doi: 10.1117/12.461507

SPIE.

Event: Astronomical Telescopes and Instrumentation, 2002, Waikoloa, Hawai'i, United States

AN ARCHITECTURE FOR ACCESS TO A COMPUTE INTENSIVE IMAGE MOSAIC SERVICE IN THE NVO

G. Bruce Berriman[†a], David Curkendall[b], John Good [a], Joseph Jacob[b], Daniel S. Katz[b], Mihseh Kong[a], Serge Monkewitz [a], Reagan Moore[c], Thomas Prince[d], Roy Williams[e]

[a] Infrared Processing And Analysis Center, California Institute of Technology
[b] Jet Propulsion Laboratory, California Institute of Technology
[c] San Diego Supercomputing Center
[d] Division Of Physics, Mathematics and Astronomy, California Institute of Technology
[e] Center for Advanced Computing Research, California Institute of Technology

Keywords: Astronomical image mosaics, image reprojection, data access, request management

ABSTRACT

The National Virtual Observatory (NVO) will provide on-demand access to data collections, data fusion services and compute intensive applications. The paper describes the development of a framework that will support two key aspects of these objectives: a compute engine that will deliver custom image mosaics, and a "request management system," based on an e-business applications server, for job processing, including monitoring, failover and status reporting. We will develop this request management system to support a diverse range of astronomical requests, including services scaled to operate on the emerging computational grid infrastructure. Data requests will be made through existing portals to demonstrate the system: the NASA/IPAC Extragalactic Database (NED), the On-Line Archive Science Information Services (OASIS) at the NASA/IPAC Infrared Science Archive (IRSA); the Virtual Sky service at Caltech's Center for Advanced Computing Research (CACR), and the *yourSky* mosaic server at the Jet Propulsion Laboratory (JPL).

1. INTRODUCTION

The National Virtual Observatory (NVO) [1] will be a new kind of observatory, one that will give astronomers access to distributed data sets and services from their desktops. Through existing astronomy portals and World Wide Web sites, astronomers will have access to powerful new data discovery and information services, to time-intensive data delivery services, and to compute-intensive services such as cross-matching between large catalogs, statistical analysis of large data sets, and generation of image mosaics of arbitrarily large area. These powerful new services must be deployed side-by-side with existing services now available through the same portals.

We are actively developing a compute-intensive services image mosaic service for deployment in the NVO. Our work concerns the development of the software itself and the broader issue of how requests for such compute intensive processing will be managed, given that the NVO architecture must support many simultaneous requests and must respond to status messages that report unfulfilled or partially fulfilled requests. We are developing the service itself and middleware whose role is to manage requests, respond to status messages and provide load balancing. This middleware will be capable of handling requests for any compute-intensive or time-intensive service, but in the first instance will manage requests to the mosaic service.

The architecture of the mosaic service is an evolution of the design of the *yourSky* service [2], described elsewhere in this volume [3]. It supports simple background removal through flattening the background in the images, and has thus

[†] gbb@ipac.caltech.edu; phone 1 626 395-1817; fax 1 626 397-7354; http://irsa.ipac.caltech.edu; Caltech Mail Stop 100-22, Pasadena, CA 91125.

far been run on an SGI Onyx platform (though written to conform to ANSI C standards). The next generation service, called Montage [4], will support the following improvements to *yourSky*:

- Greater scientific fidelity in the mosaics, such as conversation of energy in the mosaics, and support for the "Drizzle" algorithm [5]
- Application of physically based background subtraction models
- Improved performance throughput
- Interoperability with grid infrastructure
- Compliance with NVO architecture

For clarity of presentation, we will assume that the request management and Montage form a complete end–to–end system. They will be deployed as such in the initial release, but they are part of a wide effort across the NVO to develop an architecture that will support processing at scale. Thus over the lifetime of the NVO project, the services described here will be fully integrated into the high-level NVO architecture shown in Figure 1. In the context of this architecture, the request management system will sit at the top of layer 2, and will be the first layer of NVO compliant middleware that data requests encounter. The applications themselves are part of compute resources in layer 7.

This paper describes the high-level architectures of the image mosaic service and of the request management system. Section 2 describes Montage: the science goals, the architecture, and how the service will be deployed operationally on a computing grid. Section 3 describes the aims and architecture of the request management system, called the Request Object Management Environment (ROME), and how it functions operationally from a user's point-of-view. Section 4 describes how Montage and ROME work in tandem to fulfill requests for image mosaics. Section 5 describes release schedules for these services.

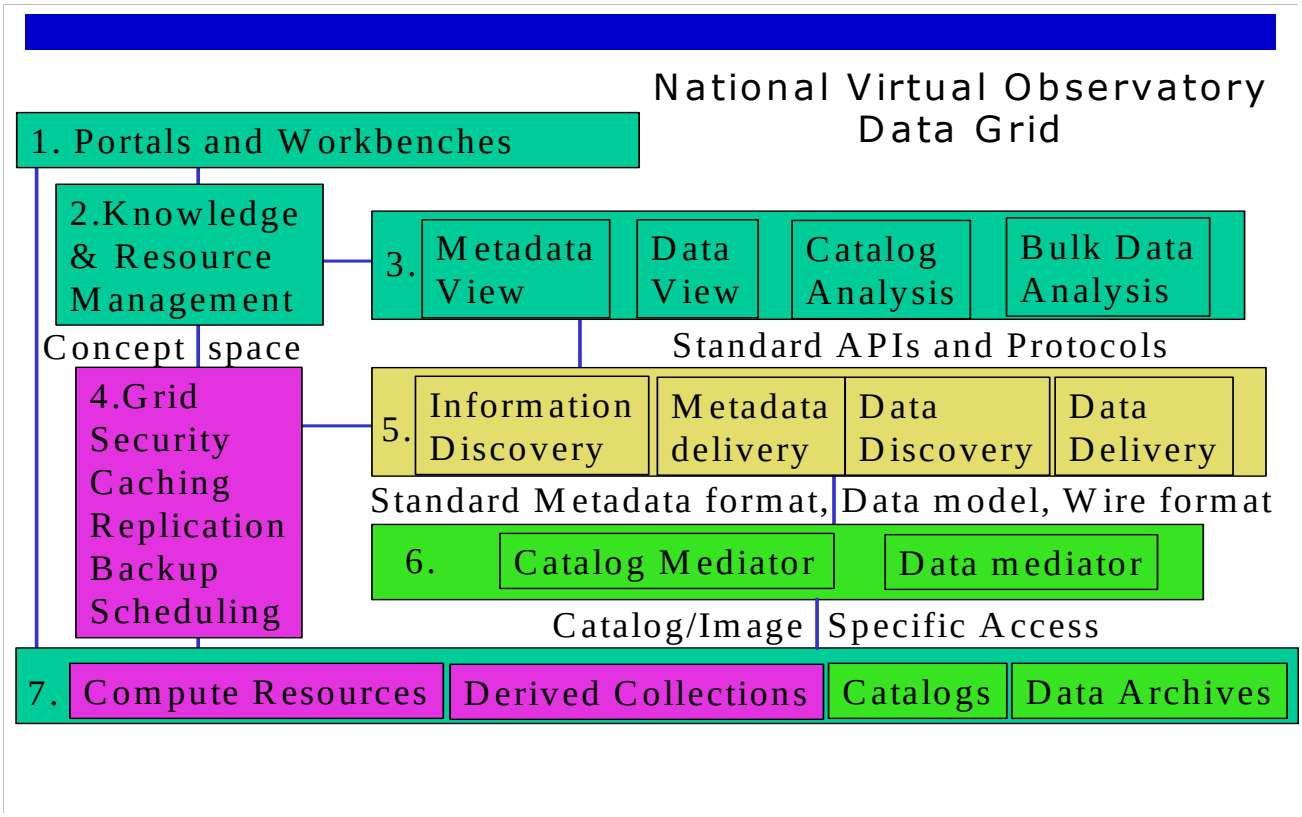


Figure 1: The High Level Design of the Architecture of the Data Grid for the National Virtual Observatory.

2. MONTAGE: AN ASTRONOMICAL IMAGE MOSAIC SERVICE

2.1 Science Goals of Montage

Astronomy has a rich heritage of discovery from image data collections that cover essentially the full range of the electromagnetic spectrum. Image collections in one frequency range have often been studied in isolation from image collections in other frequency ranges. This a consequence of the diverse properties of the data collections themselves – images are delivered in different coordinate systems, map projections, spatial samplings and image sizes, and the pixels themselves are rarely co-registered on the sky. Moreover, the spatial extent of many astronomically important structures, such as clusters of galaxies and star formation regions, are substantially greater than those of individual images.

While tools have been developed for generating mosaics, they are generally limited in scope to specific projections and small regions, or are available only within astronomy toolkits. Montage aims to rectify this unsatisfactory state-of-affairs by delivering an on-demand custom image mosaic service that supports all common astronomical coordinate systems, all WCS map projections, arbitrary image sizes (including full-sky images), and user-specified spatial sampling. This service will be available to all astronomers through existing astronomy portals, and in its initial deployment, will serve images from the 2 Micron All Sky Survey (2MASS) [6], Digital Palomar Observatory Sky Survey (DPOSS) [7] and Sloan Digital Sky Survey images (SDSS) [8]. portable version that will be available for use on local clusters and workstations will generate mosaics from arbitrary collections of images that are stored in Flexible Image Transport System (FITS) [9] format files.

Montage will deliver mosaics from multiple image data sets as if they were a single multi-wavelength image with a common coordinate system, map projection *etc*. Such mosaics will widen the avenues of research, and will enable deeper and more robust analysis of images than is now possible, including:

- Deep source detection by combining data over multiple wavelengths
- Spectrophotometry of each pixel in an image
- Position optimization with wavelength
- The wavelength dependent structure of extended sources
- Image differencing to detect faint features
- Discovering new classes of objects *etc*

2.2 The Architecture of Montage

Ground-based observatories generally correct images only for instrumental features and optical distortions. Removal of emission from the night sky is requisite to reliable astronomical analysis, and so Montage must bear the burden of removing background radiation as well as meeting the customer's image specification demands. These two needs have driven the high-level design of the Montage software, shown in Figure 2 and described in more detail in Table 1. Montage consists of two independent but interoperable components: a background rectification engine, responsible for removal of background radiation, and a coaddition/reprojection engine, responsible for computing the mosaic. Montage will support all reprojections defined in the World Coordinate System (WCS) [10].

The Montage processing paradigm consists of three main parts: reprojection of images to a common scale/coordinate system; background adjustment of images to a common flux scale and background level; and coaddition of reprojected/background-corrected images into a final mosaic. The background adjustment process involves fitting the differences between overlapping images on a local (for small mosaics) or global scale and determining the parameters for smooth surfaces to be subtracted from each image to bring them to the common scale. These parameters can either be determined on the fly or done once and saved in a database for any future mosaics done with the same images. The advantage of the former is that it allows variations in the fitting algorithms to deal with the special cases and, for small regions, will probably be more sensitive to local variations than a global fit. The advantage of the latter is that it provides a uniform view of the sky and a tested "best fit" that can be certified as such by the data provider. We plan to use both approaches, deriving and storing in a relational DBMS at least one set of background fit parameters for the full

sky for each image collection, but allowing the user the option to invoke custom background processing if they think it will provide a better mosaic for a local region

As an example, consider a 1-degree square mosaic of the Galactic Center as measured by 2MASS in the K_s band. Figure 3(a) shows an unrectified mosaic. The striped appearance arises because different scan paths were observed at different times and through different atmospheric path lengths. Figure 3b shows an example mosaic generated by the Montage prototype code: the images in Figure 3a have been background rectified to generate a seamless mosaic in Fig 3b. The image is, not, however of science grade and should be considered as a proof-of-concept.

The computational heart of Montage is the image reprojection, which takes up nearly all the compute time. The process is, however, inherently parallelizable, and can be run on however many processors are available to it. When deployed, Montage will sustain a throughput of at least 30 square degrees (e.g. thirty 1 degree x 1 degree mosaics, one 5.4 degrees x 5.4 degrees mosaic, etc.) per minute on a 1024 x 400 MHz R12K Processor Origin 3000 *or machine equivalent*.

Component	Description
Mosaic Engine Components	
mImgtbl	Extracts the FITS header geometry information from a set of files and creates an ASCII image metadata table from it used by several of the other programs.
mProject	Reprojects a single image to the scale defined in a pseudo-FITS header template file (an ASCII file with the output image header lines, but not padded to 80 characters and with new lines at the end of each line). Actually produces a pair of images: the reprojected image and an "area" image consisting of the fraction input pixel sky area that went into each output pixel.
mProjExec	A simple executive that runs `mProject` for each image in an image metadata table.
mAdd	Coadd the reprojected images using the same FITS header template and working from the same `mImgtbl` list.
Background Rectification Components	
mOverlaps	Analyze an image metadata table to determine a list of overlapping images.
mDiff	Perform a simple image difference between a single pair of overlapping images. This is meant for use on reprojected images where the pixels already line up exactly.
mDiffExec	Run `mDiff` on all the pairs identified by `mOverlaps`.
mFitplane	Fit a plane (excluding outlier pixels) to an image. Meant for use on the difference images generated above.
mFitExec	Run `mFitplane` on all the `mOverlaps` pairs. Creates a table of image-to-image difference parameters.
mBgModel	Modeling/fitting program which uses the image-to-image difference parameter table to interactively determine a set of corrections to apply to each image to achieve a "best" global fit.
mBackground	Remove a background from a single image (planar has proven to be adequate for the images we have dealt with).
mBgExec	Run `mBackground` on all the images in the metadata table

Tabl1 1: The Design Components of Montage

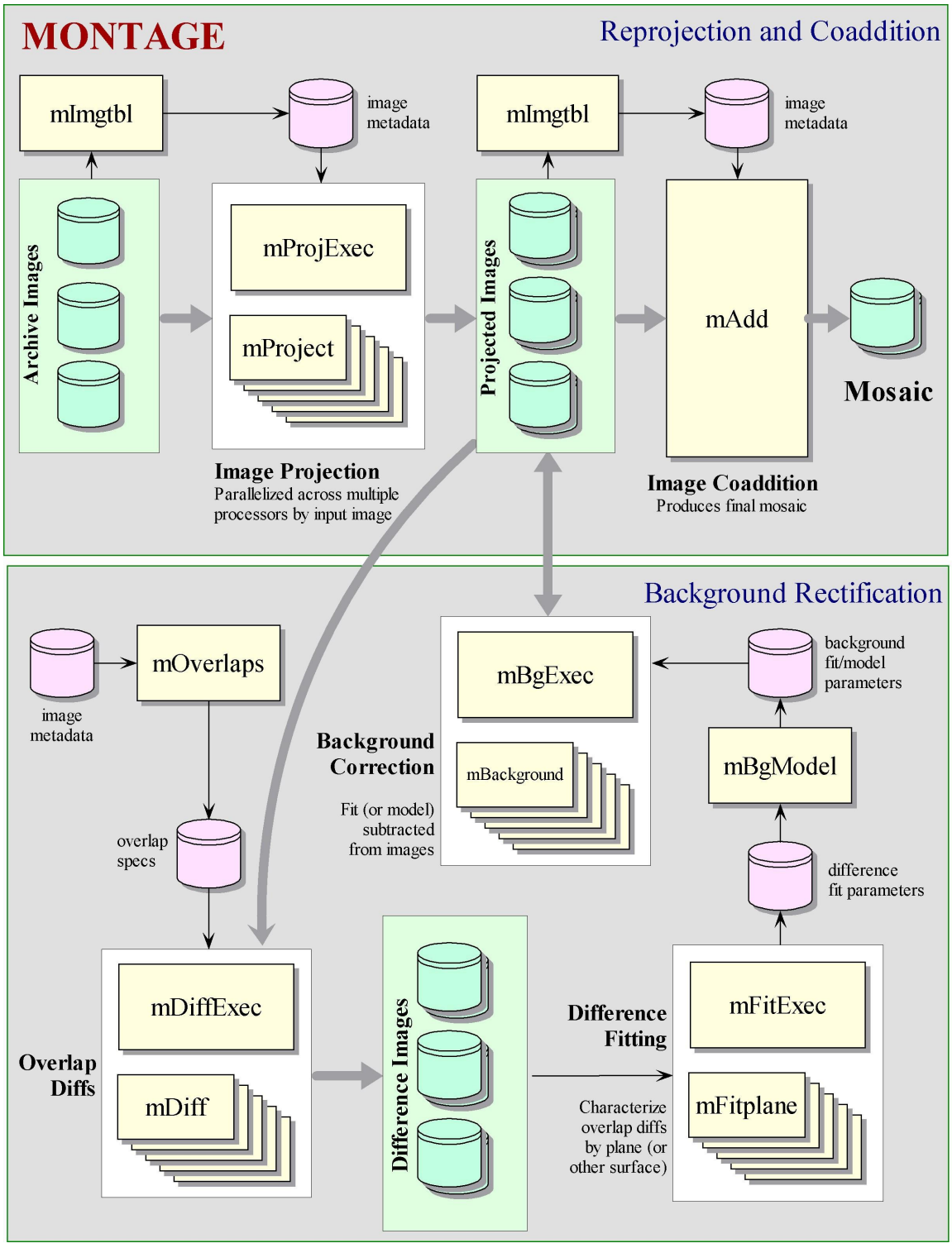
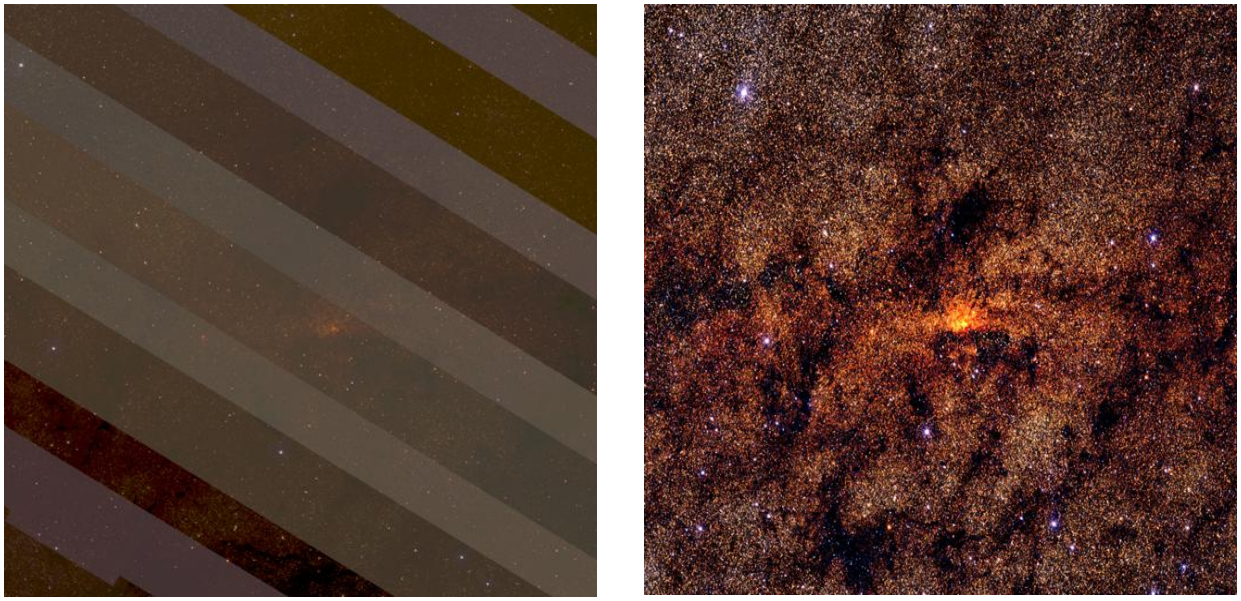


Figure 2: The high-level design of Montage. The figure shows the background rectification engine, and the reprojection and co-addition engine. The components of each engine are shown.



(a) **(b)**

Figure 3: A 1 degree square mosaic of the Galactic Center in the K band, constructed by Montage from images released by the 2MASS project. Frame (a) does not rectify the images for background emission, and the variation of background radiation at different times and air masses is apparent in the stripes. Frame (b) shows the same region after applying a background removal algorithm as described in the text. The mosaic is a demonstration product generated with a prototype code; it is not a science grade image and it is not endorsed by 2MASS.

3. THE DESIGN OF THE REQUEST MANAGEMENT SYSTEM

3.1 Why Do We Need Request Management?

A fact of life in using the World Wide Web is that the data requests can take a long time, because network bandwidth is limited or remote servers are slow. Many services overcome this fundamental limitation by passing the request to a remote server, returning an acknowledgement and later reporting the results to the user through email. This approach is used by the NASA/IPAC Infrared Science Archive to process bulk requests for browse (compressed) and Atlas (full spatial resolution) images from the 2MASS project. 2MASS is a full-sky survey in the near-infrared with uniform calibration quality. Its observational phase, now complete, produced an Atlas containing 10 TB of images. Some 4 TB of data, covering 47% of the sky, are served publicly through the NASA/IUPAC Infrared Science Archive [11], but the files themselves, 1.8 million of them each covering 0.15^o x 0.3^o of the sky, are stored on the High Performance Storage System (HPSS) [12] at San Diego Supercomputer Center. A lossy-compressed set of images, suitable for quick look, reside on a server at IRSA. The user requests images – Atlas or compressed - through a simple web-form. Server side software locates the data files that meet the request, sends a request to the appropriate server for these data, packages the files and sends them to staging area before sending a results email to the sender. The service serves over 40 GB of data each month, and at peak usage has successfully processed over 1,000 requests per day.

While the bulk image service has shown how mass storage systems and grid technology can serve the needs of astronomy, it has exposed the limitations of the approach of passing the request to a remote server. The approach is based on an act of faith that the submitted request will be filled. Users have no mechanism for monitoring the request, or

for automatically resubmitting it if the service is down. They must wait until notified electronically of a failed request, and then resubmit the request. Perhaps a more serious limitation is that the system has no load balancing mechanism: large numbers of requests sometimes grind the service to a halt.

With the deployment of the NVO framework, we anticipate ever growing use of requests for large data sets and they will place yet more strain on resources and exacerbate the frustrations described in the last paragraph. How can this problem be overcome? A simple method is to provide a mechanism for managing requests from users, responding to messages from services, and for performing simple load balancing. This section describes the design of such a system.

3.2 Overview of the Request Object Management Environment (ROME)

Broadly speaking, ROME is simply a lightweight set of tools that handle requests, respond to messages and manages pools of requests in a fault tolerant fashion. These capabilities offer obvious benefits to users and service providers. By design, ROME has no control over processing, and is intentionally decoupled from all processing details. It accepts information through HTTP connections and sends asynchronous text messages in XML format to clients through standard socket protocols. The NASA/IPAC Infrared Science Archive (IRSA) [13] has already used this type of interface to deploy a simple "message aware" client interface that shows the results of very short jobs, mainly downloads of images from remote sites. This design allows ROME to support clients in a platform and language independent fashion, and will allow it to support existing functions, largely web-based services. Thus, it will support the 2MASS bulk image service, described earlier, in way that is transparent to the user, and will support new services such as Montage as they are deployed.

3.3 High Level Design of ROME

Figure 4 shows the design of ROME (the numbers in the diagram refer to steps in the processing flow, and are described in the next section). It is built with Enterprise Java Bean (EJB) [14] e-business technology, coupled with a robust Informix Data Base Management System infrastructure. The EJB framework is an industry standard for business component development, and resides in a specialized JAVA VM (EJB server). EJB's are components that manage and persist transactions, and handle concurrency and security.

Typical business use cases involve millions of very quick transactions, such as cash requests at an Automated Teller Machine, where EJB technology relieves the developer of direct responsibility for load management. The application programmer develops EJBs to represent data records in a database and EJBs to perform transactions — the EJB server manages the rest, and all information processing is conveniently handled within the EJB code itself. The first release will use a commercial EJB server, BEA WebLogic™ [15], but later releases will use an Open Source product. Astronomy's needs are different from those of a bank in that many queries will be long lived, and this governs a fundamental aspect of the design made clear in Figure 4: the component managing requests, the "Request Manager", is separated from the component that processes the requests, the "Request Processor".

The Request Manager (RM) is simply a collection of persistent EJB-based services whose only function is to manage knowledge relating to user requests for processing. Requests are submitted through standard client interfaces, such as Web forms or JAVA GUIs, which collect user parameters and submit requests to a standard HTTP Web server interface. The Request Manager persists information about requests in the database, in case of application or system failure, and can easily support thousands of simultaneous requests. The Request Processor (RP) is an engine for overseeing load balancing and conveying status information back to the Request Manager (and ultimately the user). Load balancing is achieved by having several request processors external to the RM, each of them having multiple "worker threads" to run the applications that actually process the query. Each request is a separate thread that gives the user control over the process. For example, based on the results of a quickly-returning thread, the user may decide that he or she does not need to complete some other job and can send an interrupt to abort it. Similarly, after monitoring long-running requests for a while, he or she may decide to add email notification and disconnect.

3.4 A Typical ROME Processing Scenario

Figure 4 illustrates at least one instance of all the components that interact with ROME. This section describes, mainly from the user's point of view, the steps in a typical processing scenario, numbered in sequence in the figure:

(1) A user submits a query,
(2) A worker thread in the RP picks up the request,
(3) The worker thread starts a copy of a processing application,
(4) The worker thread sends information about itself to the RM,
(5) The worker thread sends input parameters to the application,
(6) The application sends messages to the worker thread,
(7) The worker thread sends message to the RM,
(8) The RM sends message to the user,
(9) The user queries the RM for information pertaining to a certain request.

Users submit a request via an HTML GET/POST request, and the RM records the information in the database (Step 1). Processor threads in the RP will find the request in the database – after completing existing requests if necessary - and submit it (Step 2). The worker thread in the RM finds the request by making an HTTP call to the RM, and then processes the request by running a CGI application called with the Java runtime exec() method. This application could be a simple C program or a complex processing request. The worker thread sends the application process ID to ROME (Step 4) so that the RM can interrupt the job at any time (usually based on a user request). The application `stdin/stdout` streams are connected to pipes in the worker thread (Steps 5 and 6) and all application messages are sent through these pipes. "Results" data are usually staged to a disk "workspace", and are made accessible via a URL reference. When there is anything to report (messages from the application, completion status, *etc*.), the RP worker thread sends it to the RM, again through an HTTP interface (Step 7).

If the RP receives an interrupt from RM (through the socket mentioned earlier), it uses an operating system signal mechanism to interrupt the application program. The application is responsible for responding to the interrupt with any action (such as shutting down) and messages (such as "`DONE/Interrupted`") that are appropriate.

A JAVA client GUI (or any other client) can take full advantage of ROME capabilities to monitor and control the users' jobs, whether submitted by custom clients or through Web forms. Not only can ROME return status to clients when requested (polling), but it can send asynchronous messages directly (Steps 8 and 9). When the user starts a client, the client sets up a port to listen for messages and contacts ROME with the user's ID (email address) and this port information. ROME then sends all new (and outstanding old) messages to this location. The user can change clients, machines or ports at any time (log off, go home, and reconnect from there). The system could even send messages to a collection of locations though this will not be implemented in the first release.

The 2MASS bulk image service described above emits intermediate messages (*e.g.* the location of each individual file as it retrieves it). The client environment now not only gets these messages but can use the information to initiate intermediate file transfers. The user may even decide to abort the process midway.

There is one special connection mechanism out of the Request Manager. When messages are available for specific users about their jobs, a client program can register to receive them asynchronously. It does so by specifying a port (on the client), which is set to listen for simple message (*e.g.* via Simple Object Access Protocol or equivalent industry standard protocol). The RM will attempt to contact any such sockets that are registered (Step 8). If it fails to connect, it will delete the registration and hold all messages until the next time the user client connects (or until a timeout tells it to send email instead).

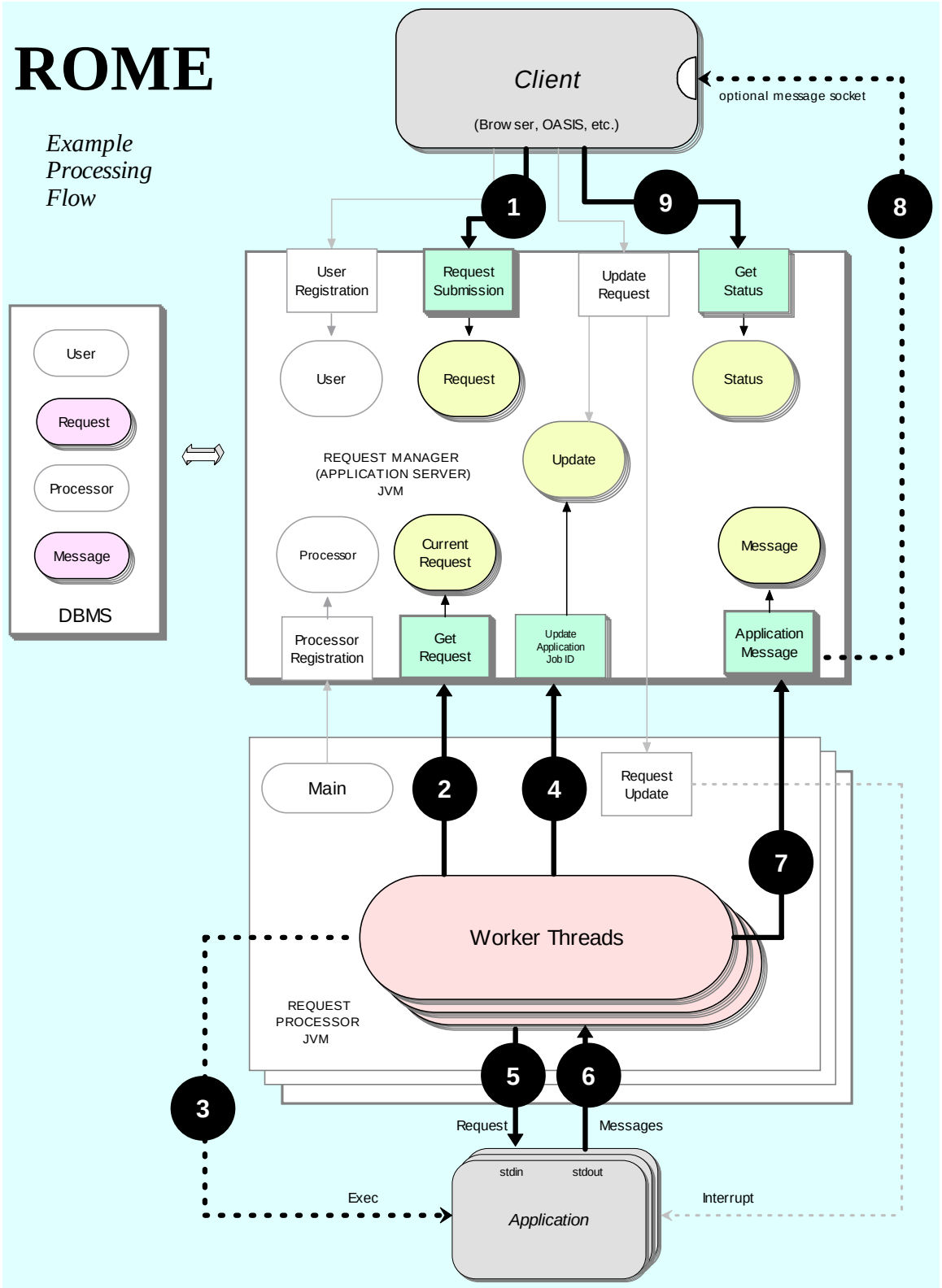


Figure 4: High Level Design of ROME, and Processing Steps (1 through 9).

4. ROME AND MONTAGE WORKING IN TANDEM

Montage will run operationally on the Teragrid [16], a high performance computational grid provided by the NSF Partnership for Advanced Computational Infrastructure. The Teragrid provides aggregate computational power on the order of 10 Teraflops, aggregate disk cache on the order of 800 TB and archival storage capacity of 6 Petabytes.. The details of how NVO compliant processes will be authenticated and fulfilled under the Teragrid are under development, but will follow the grid paradigm, where data needed for the request are obtained from the most convenient place, and computing is done on any available platform where the request can be authenticated.

A request to Montage must be satisfied transparently: users will only be aware that they are requesting an image mosaic according to their specification of position, size, projection *etc*. They will not be aware of where the request is performed, or if the image can be delivered or subset from a cached file. Figure 5 shows how a request to Montage will be handled when the architecture is fully deployed. The request is passed from the client to ROME, where it is registered in the database and submitted by the Request Processor for processing on the Teragrid. The job will be built on the Teragrid with standard Grid technologies such as the Globus [17], an Open Source toolkit that handles the construction and management of Grid processes, security *etc*. Part of the request may already be satisfied in cached image mosaics. The cache will actually be part of a data management system that subsets files and constructs new mosaics from subsets, as needed. Montage will therefore search through a catalog of cached images and will satisfy such parts of the request as it can from the cached images. If cached files cannot fill the request, processing on the Teragrid will fill it. An interpreter accepts the XML request for the Request Processor, and translates it into a suitable computational graph (directed acyclical graph, DAG) that specifies the computations that are needed and what data are needed. The DAG represents the sequence of computations needed to construct the mosaic from the input data. Montage will also perform a spatial search on the image collection metadata to find those files needed to fill the request. The data themselves will reside on high-quality disks, with high throughput I/O to the Teragrid processors that will be used by NVO services.

The result of the processing will be conveyed to the user through ROME. The user will receive a message that the data are available for pick-up until a deletion date. If the request was time intensive, the user may have logged off the portal and decided to wait for email notification. If the request could not be processed, ROME will be able to restart the job on the user's behalf. If only some intermediate products could be processed before the server failed, ROME will rerun the job, but find the intermediate products and use them as inputs. Many other partial processing examples can be handled easily within ROME.

5. DELIVERY SCHEDULES

Montage will be fully operational in January 2005. The first public release of the code will be in February 2003, which will emphasize accuracy in mosaics over performance. Subsequently, there will be two deliveries per year with progressive in improvements in functionality and performance. A prototype version is currently available to users on a shared risk basis. Interested parties should contact the Montage project (montage@ipac.caltech.edu).

The ROME and Montage deliveries are independent of each other. We are currently developing the initial release of ROME, which is scheduled for release in September 2003, but will use a commercial Applications Server, BEA Web Logic. Updated releases, and a version using an Open Source Applications Server, will be released at yearly intervals for the duration of the NVO project.

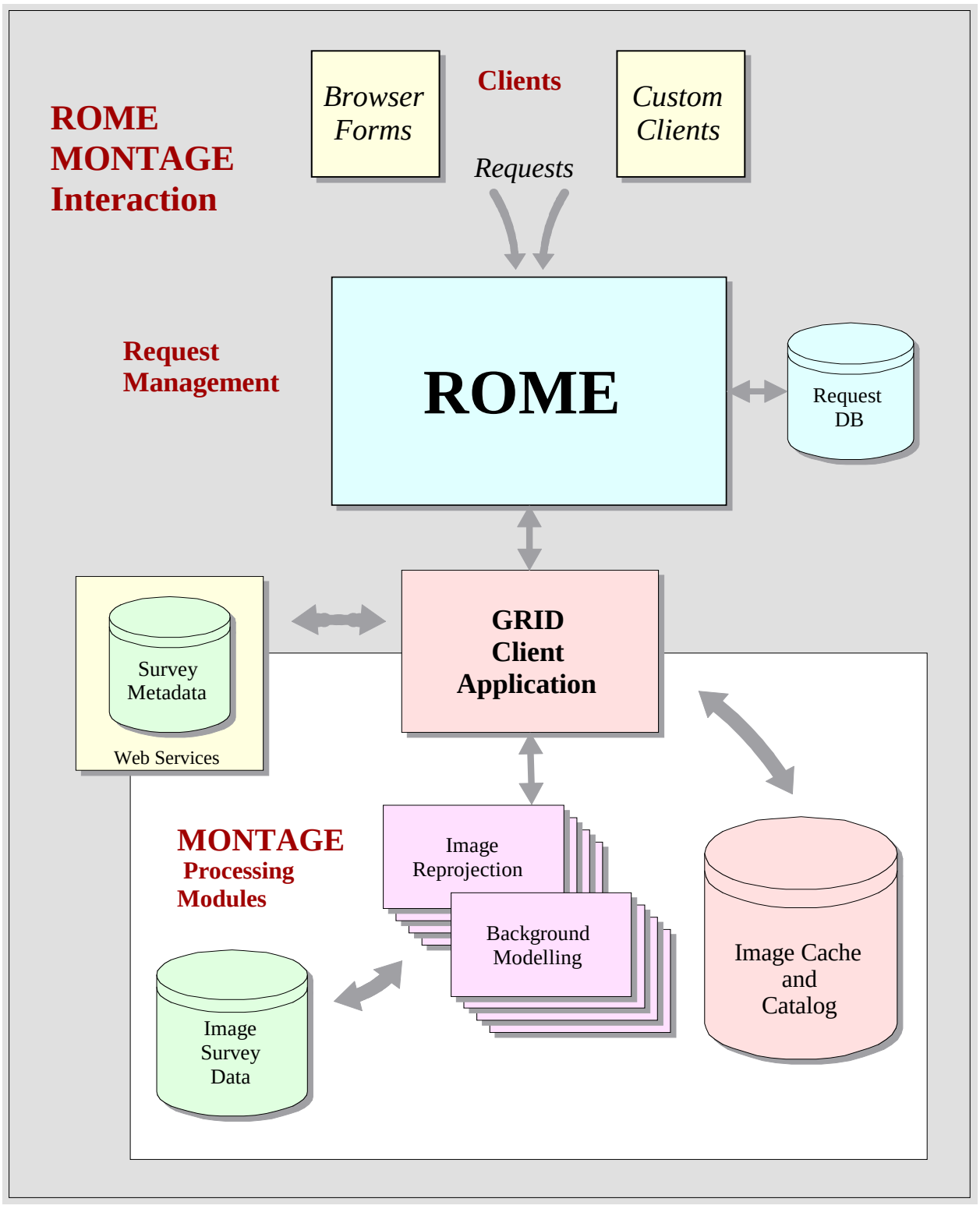


Figure 5: The Architecture of ROME and Montage In Operations Under the NVO

ACKNOWLEDGEMENTS

The request management system is based upon work supported by the National Science Foundation under Cooperative Agreement No. AST-0122449. Montage is supported by the NASA Earth Sciences Technology Office Computing Technologies program, under Cooperative Agreement Notice NCC 5-6261.

We wish to thank Dr. Ewa Deelman, Dr. George Kremenek, and Dr. Joseph Mazzarella for constructive discussions.

REFERENCES

1. The National Virtual Observatory (USA), http://us-vo.org.
2. yourSky, http://yourSky.jpl.nasa.gov
3. J.C. Jacob, R. Brunner, D. C. Curkendall, S. G. Djorgovski, J. C. Good, L. Husman, G. Kremenek, and A. Mahabal. "yourSky: Rapid Desktop Access to Custom Astronomical Image Mosaics," to appear in Astronomical Telescopes & Instrumentation: Virtual Observatories, SPIE (this volume)
4. Montage, http://montage.ipac.caltech.edu/
5. A.S. Fruchter, and R.N. Hook. "Linear Reconstruction of the Hubble Deep Field," http://www.stsci.edu/~fruchter/dither/drizzle.html
6. The 2MASS Project, http://www.ipac.caltech.edu/2mass
7. The Digitized Palomar Observatory Sky Survey (DPOSS), http://www.astro.caltech.edu/~george/dposs
8. The Sloan Digital Sky Survey, http://www.sdss.org/
9. The Flexible Image Transport System (FITS), http://fits.gsfc.nasa.gov, http://www.cv.nrao.edu/fits.
10. E.W. Greisen and M. Calabretta, *Representation of Celestial Coordinates In FITS*, http://www.atnf.csiro.au/people/mcalabre/WCS.htm.
11. The NASA/IPAC Infrared Science Archive 2MASS Batch Image Server, http://irsa.ipac.caltech.edu/applications/2MASS/RegionData
12. High Performance Storage System (HPSS), http://www.cacr.caltech.edu/resources/HPSS/index.html.
13. The NASA/IPAC Infrared Science Archive, http://irsa.ipac.caltech.edu/
14. Enterprise Java Beans, http:/java.sun.com/ejb/
15. BEA WebLogic™ Applications Server, http://www.bea.com/
16. The Terascale facility, http://www.teragrid.org/
17. The Globus toolkit, http://www.globus.org/